وَمَا أُوتِيتُمْ مِنَ الْعِلْمِ إِلَّا قَلِيلًا

Analog Integrated System Design
Lab 04 (Mini-Project Part 1) – xschem/ngspice
SAR ADC

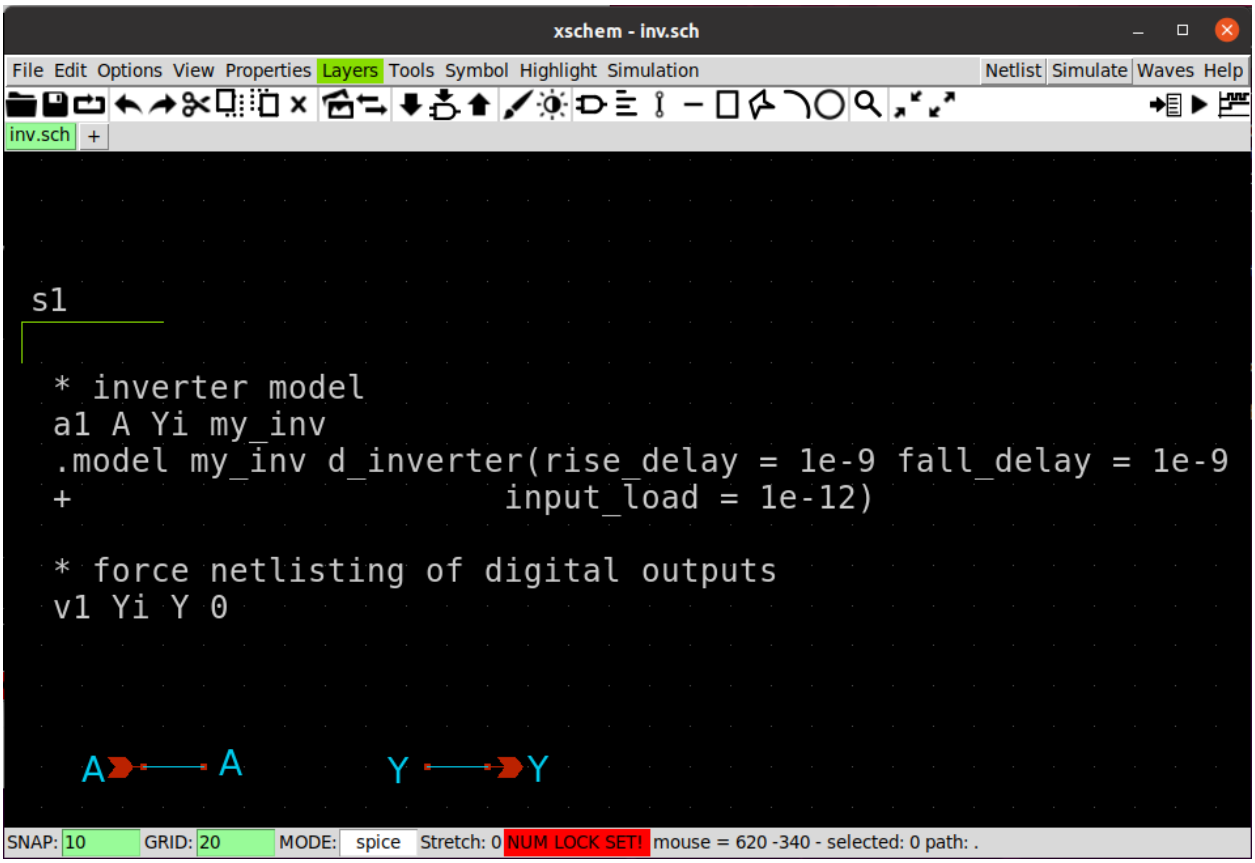
Intended Learning Objectives

1) To be familiar with behavioral modeling of mixed-signal and digital blocks.
2) To be familiar with the design and simulation of SAR ADC.

Pre-Lab: Behavioral Models

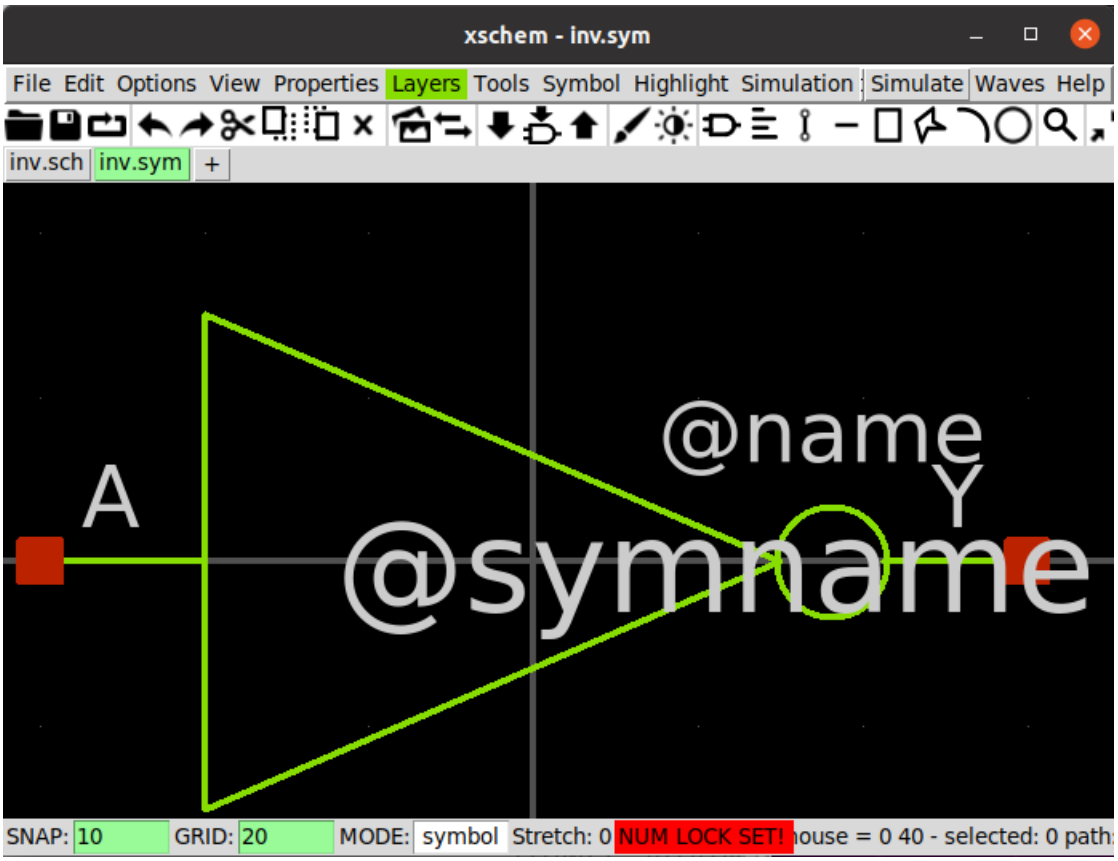
In this prelab, we will simulate behavioral models of some basic components. Eventually, all components must be built using transistors. But behavioral models can be useful in the initial phase of any project to build and verify a system quickly.

1) Create a new work directory “sar_adc”. Place all project files under this work directory.
2) In this lab we will deal with transistors from GF180 PDK. Copy the file “/home/tare/XschemForGF/xschemrc” to your work directory. This file initializes xschem for usage with GF180.
3) Open the terminal in your work directory. Run the commands below in the terminal:
4) $ export PDK_ROOT=/home/tare/pdk
5) $ export PDK=gf180mcuC
6) Open xschem and create a new schematic from the terminal (ignore the warning that this file doesn’t exist)”
   xschem inv.sch
7) Read “Sec 12.4 Digital Models” in ngspice documentation:
   https://ngspice.sourceforge.io/docs/ngspice-html-manual/manual.xhtml
8) Create a schematic for the inverter.

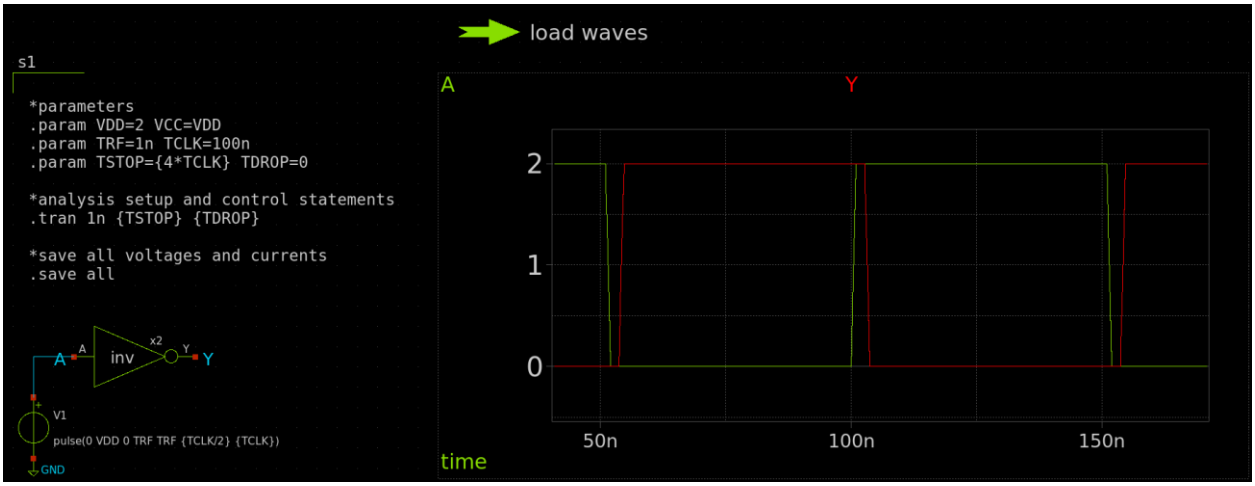



```
name=s1 only_toplevel=false
value="
* inverter model
a1 A Yi my_inv
.model my_inv d_inverter(rise_delay = 1e-9 fall_delay = 1e-9
+                      input_load = 1e-12)

* force netlisting of digital outputs
v1 Yi Y 0
"
```

9) Create a symbol for the inverter (Symbol -> Make symbol from schematic). Edit the symbol as shown below.



10) Build a testbench as shown below to verify the inverter model.
Note: The VCC param must be defined to specify the supply voltage of the digital models.

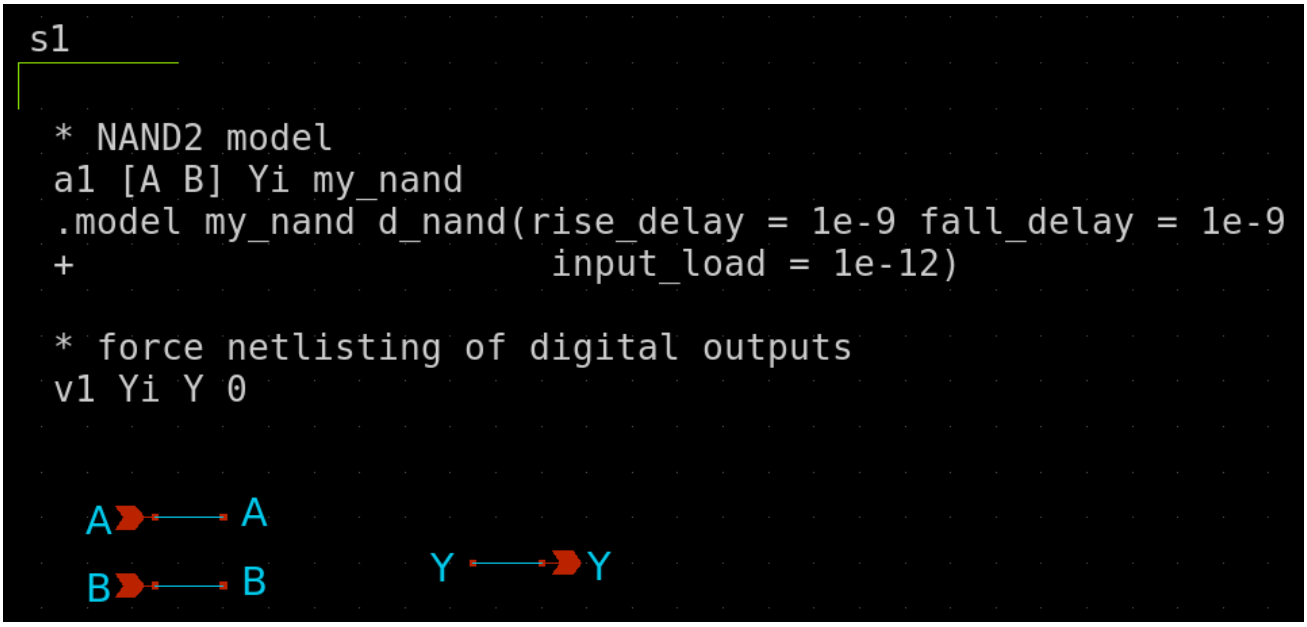


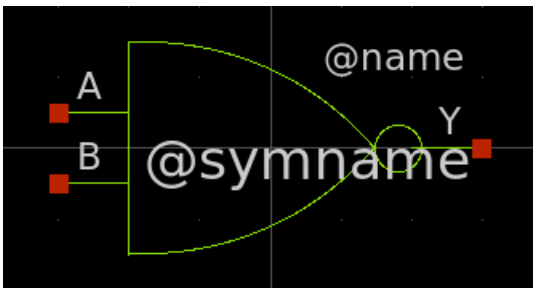

```
name=s1
only_toplevel=false

value="
*parameters
.param VDD=2 VCC=VDD
.param TRF=1n TCLK=100n
.param TSTOP={4*TCLK} TDROP=0

*analysis setup and control statements
.tran 1n {TSTOP} {TDROP}
```

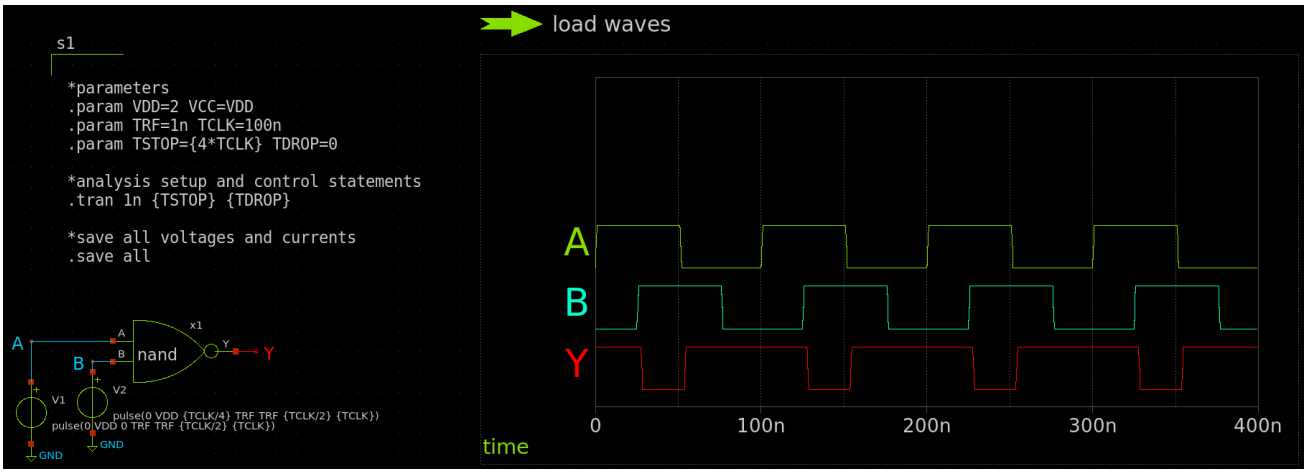
```
*save all voltages and currents
.save all
"
```

11) Repeat the previous steps to verify the components below. Create a new schematic and symbol for each component. Create and modify the testbench as needed to verify the operation.
    - d_nand
    - d_nor
    - d_dff

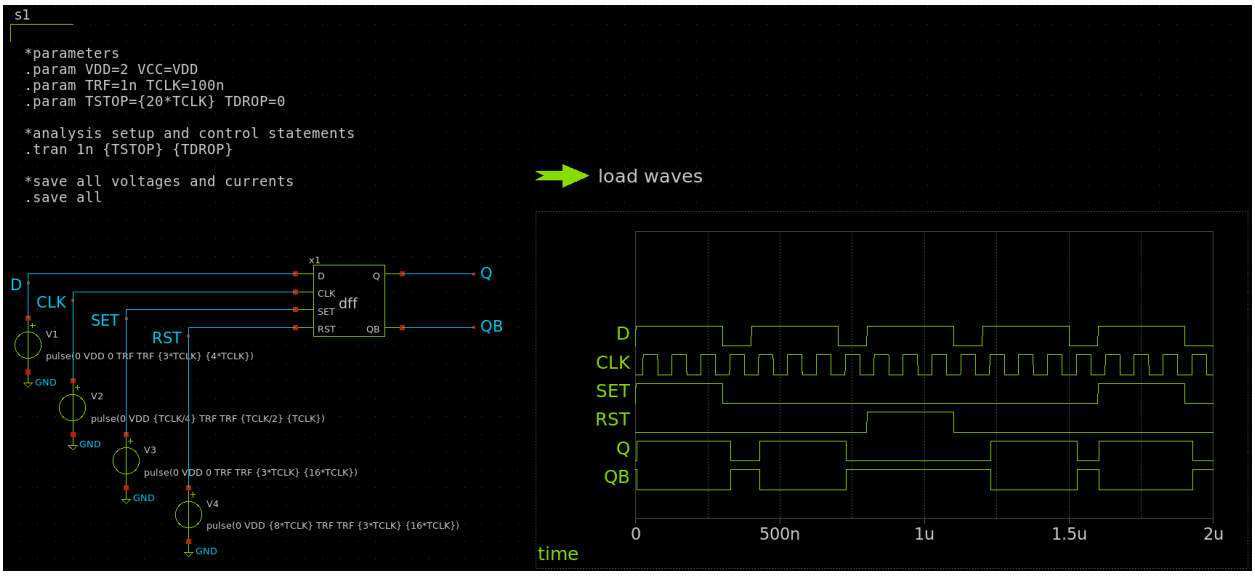
12) Note that the testbench must test all the possible cases. Below are examples for a NAND2 gate.



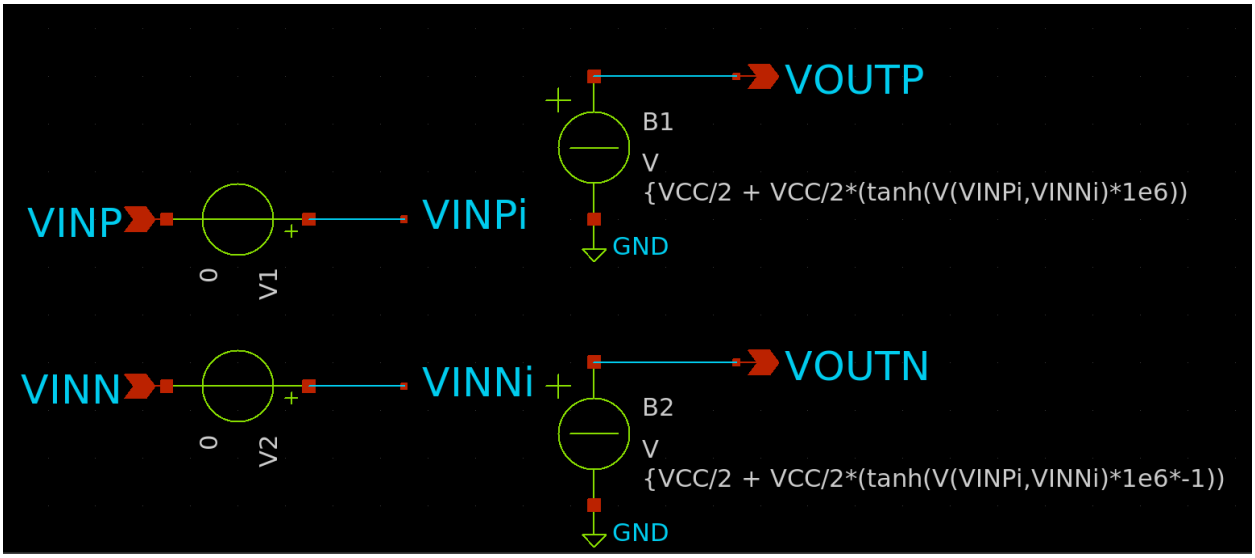




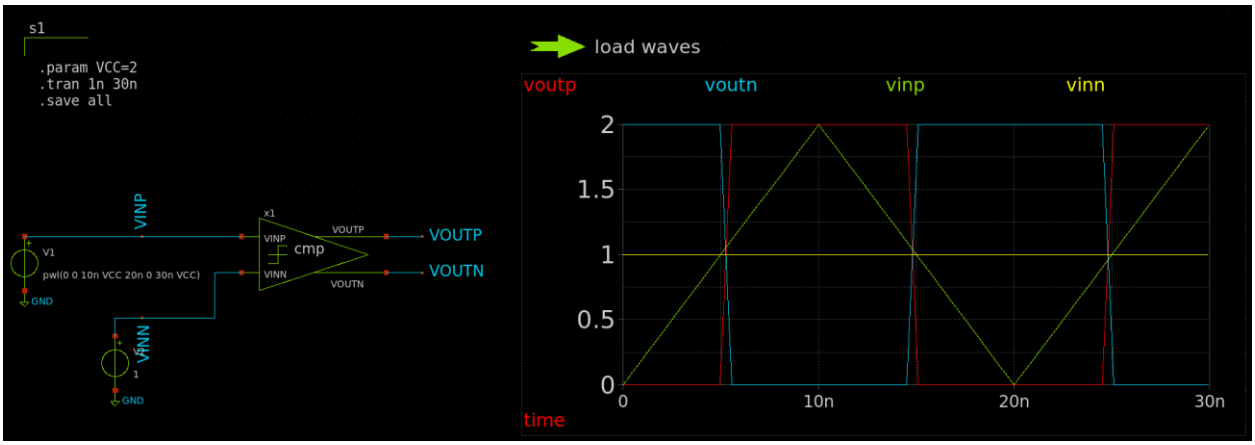
13) This is an example of a good testbench setup for the D-flipflop.



14) Create a behavioral model for a comparator circuit. In the model below, bsource (from xschem_library/devices) is used for the output and a zero voltage source is used to force the netlisting of the input pins.



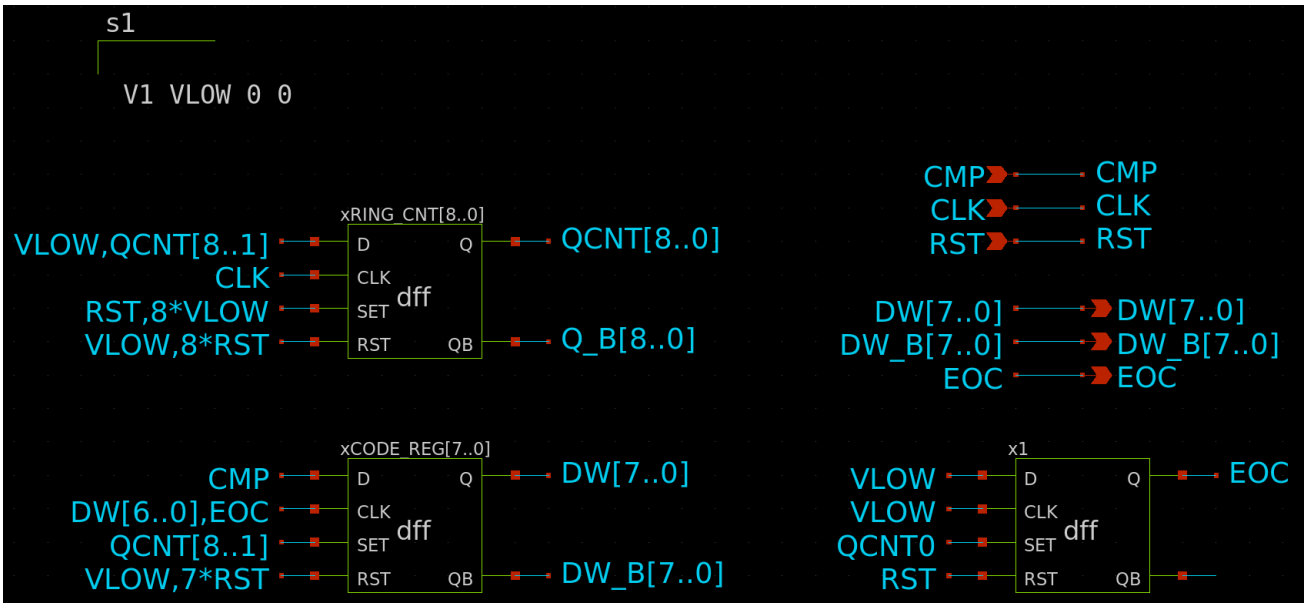
15) Create a symbol for the comparator (Symbol -> Make symbol from schematic). Build a testbench to verify it.
Note: You can use a DFF after the comparator to turn it into a dynamic (clocked) comparator.



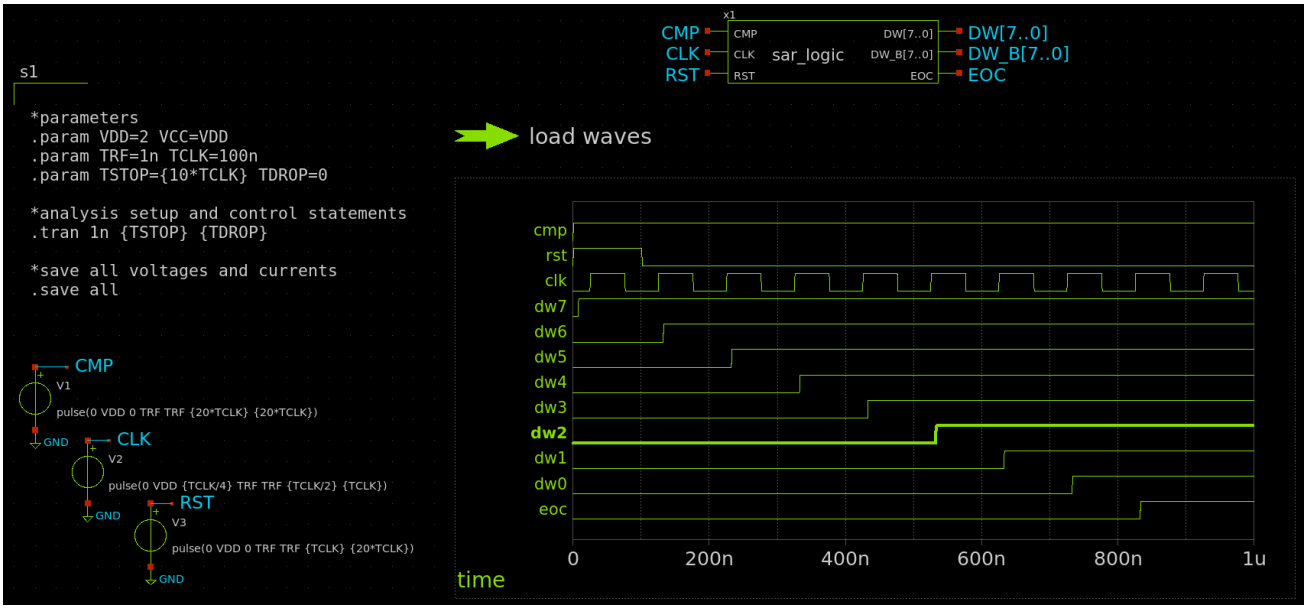
16) Report transient simulation results showing proper operation of each component.

PART 1: SAR Logic

1) Create the schematic of SAR logic as shown below. Study the operation carefully.



2) Create a testbench to verify SAR logic operation. Set FCLK = (NBIT + 2) * FS, where NBIT = 8 and FS = 1MHz.



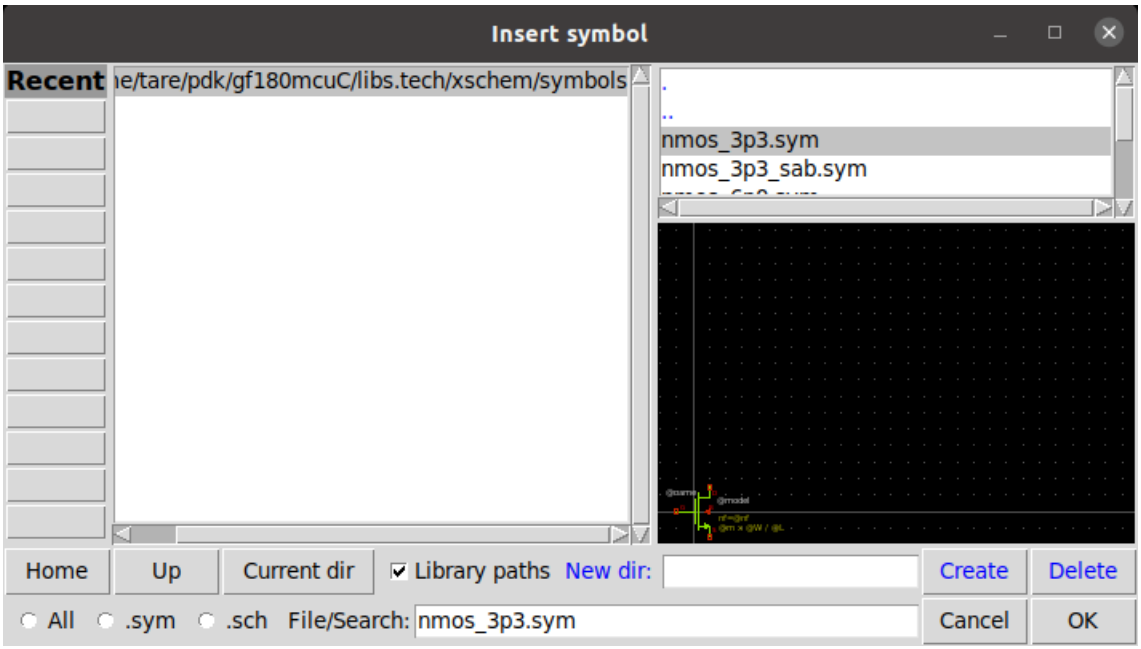
3) Report transient simulation results for the ring counter output, the code register output, and the EOC signal for the following cases:
   a. CMP is all zeros
   b. CMP is all ones
   c. CMP is alternating ones and zeros (period = 2*TCLK and pulse width = TCLK)
4) Briefly explain **in your own words** how does this design work and how does it implement the successive approximation algorithm[1]. Support your explanation by filling the state table below.

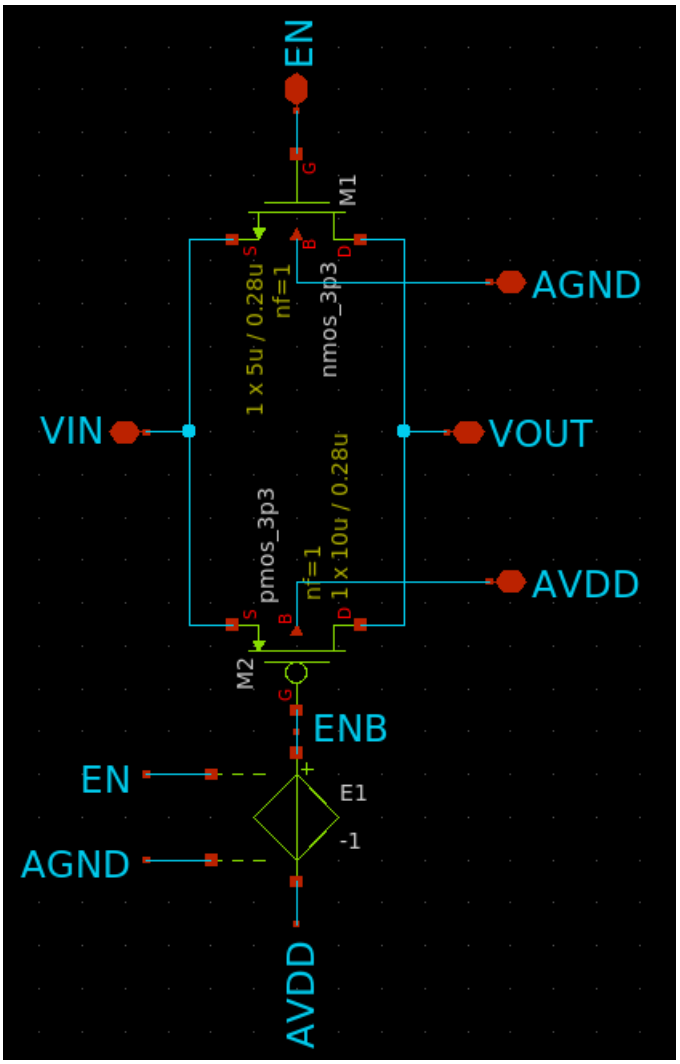
Clock cycle	DW<7>	DW<6>	DW<5>	DW<4>	DW<3>	DW<2>	DW<1>	DW<0>	CMP
1 (reset)									
2									B7
3									B6
4									B5
5									B4
6									B3
7									B2
8									B1
9									B0
10									

PART 2: Transmission Gate

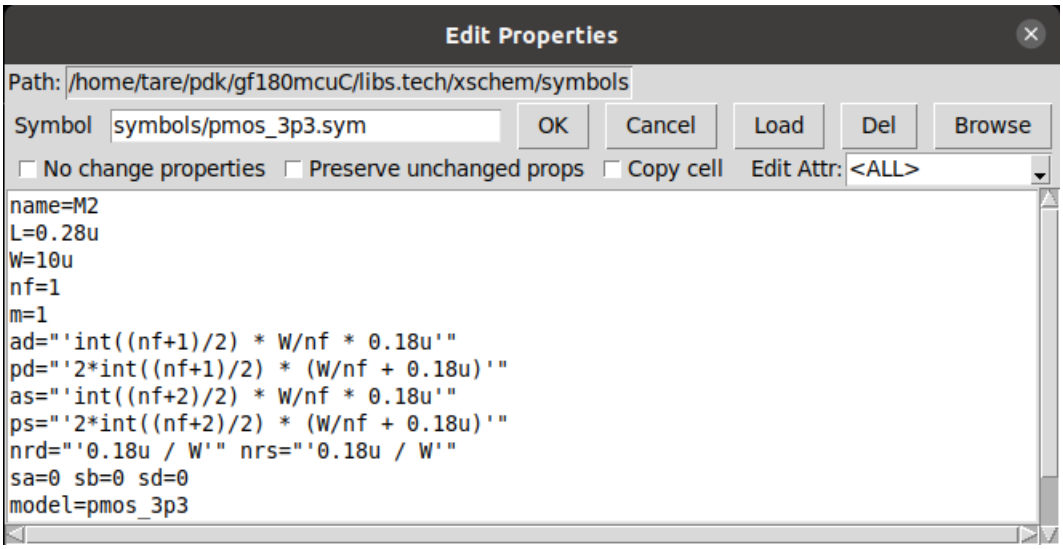
1) Create a new schematic (File -> Create new window/tab).
2) Save the schematic in the current directory (ctrl+s). Name it “tg.sch”.
3) Create a CMOS transmission gate (TG) as shown below. The VCVS is used as ideal inverter to generate the ENB signal.
4) For the NMOS and PMOS use nmos_3p3 and pmos_3p3 from GF180 library as shown below (gf180mcuC/libs.tech/xschem/symbols).
   Hint: Search for the required symbol using the Search field. Use Alt+R to rotate the symbol.

[1] You may find this reference useful (but be aware of typos and mistakes): Hedayati, Raheleh. "A study of Successive Approximation Registers and implementation of an ultra-low power 10-bit SAR ADC in 65nm CMOS technology." (2011).

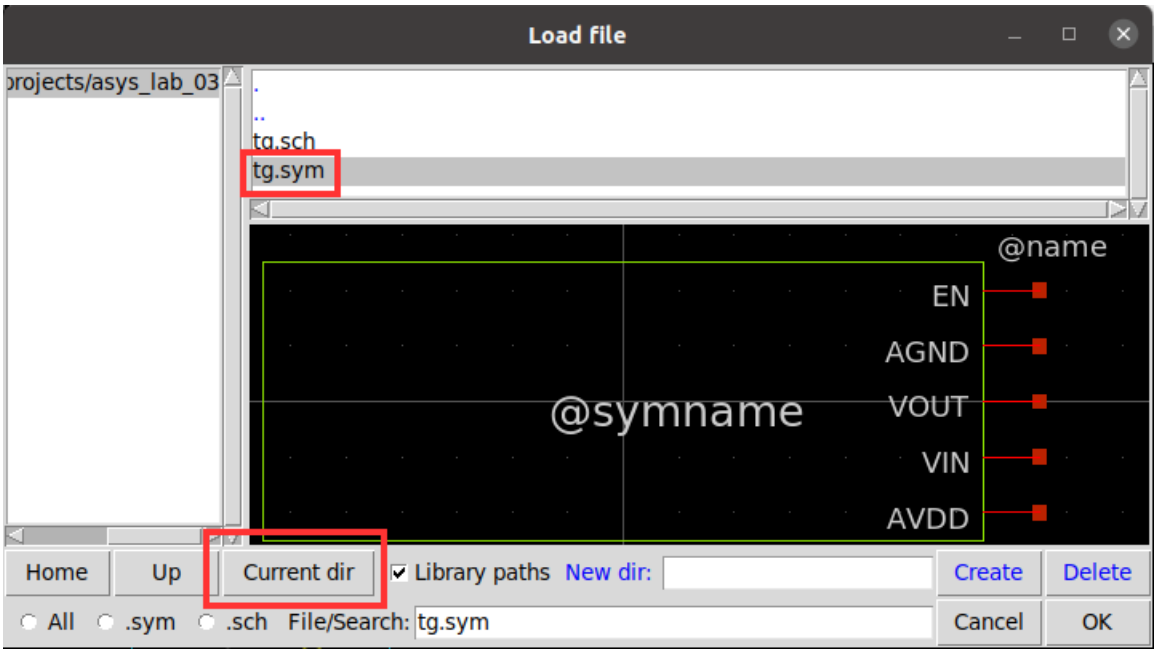




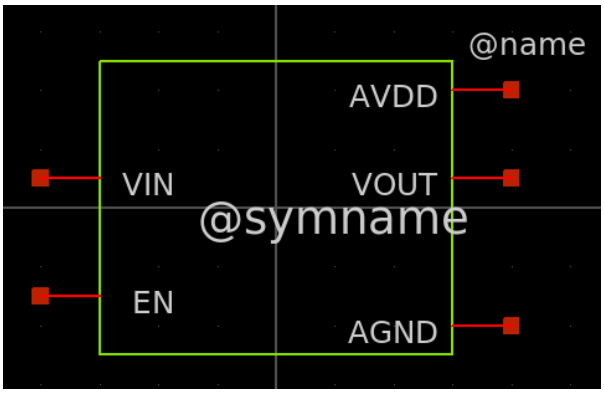
5) Edit the transistor properties and set W of NMOS to 5u and W of PMOS to 10u.



6) Save the schematic. Create a symbol for the TG: Symbol -> Make symbol from schematic, or use the hotkey ‘a’.
7) Open the symbol file: File -> Open.

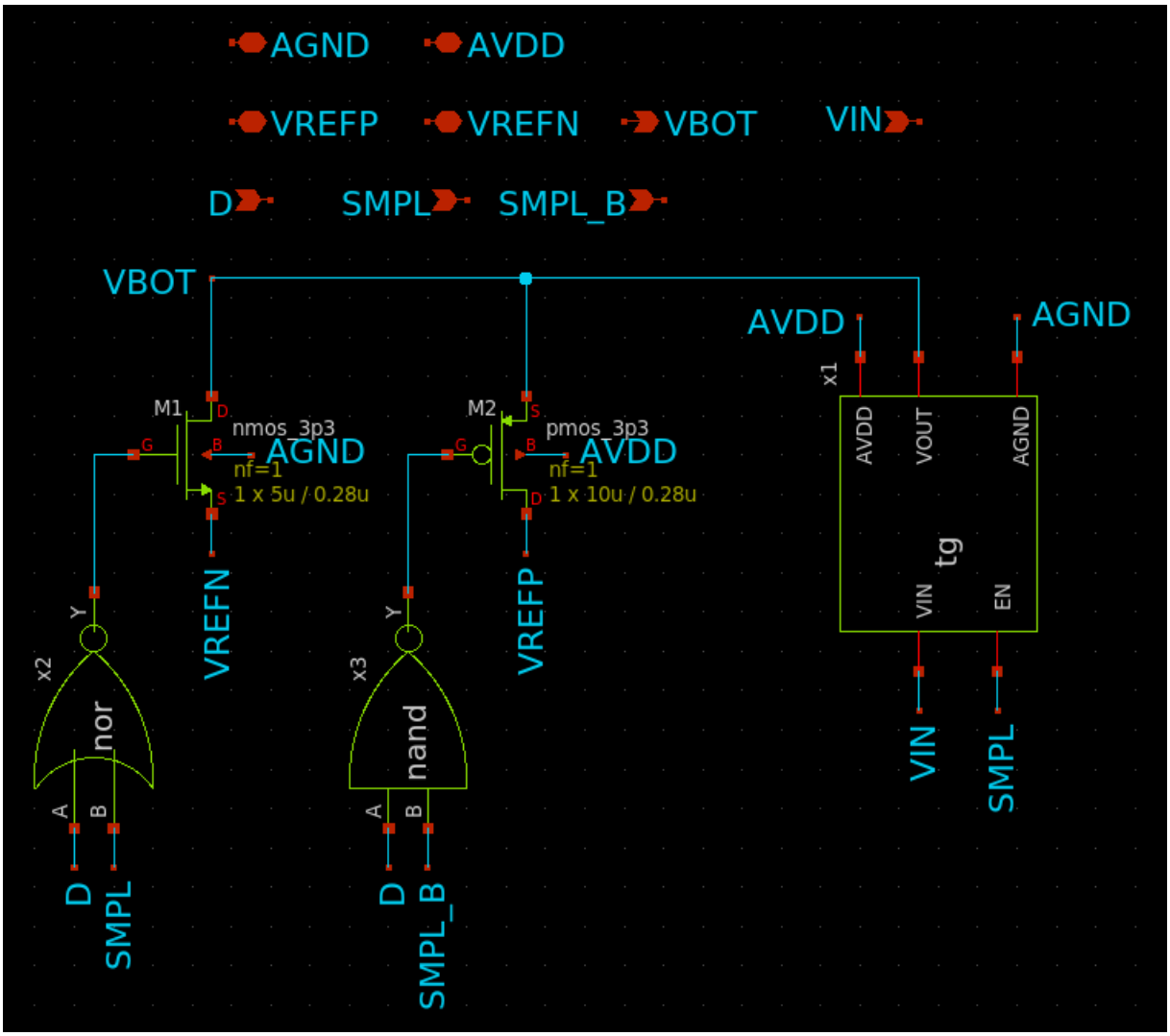


8) Edit the symbol to look as shown below. Use the following hotkeys: ‘m’ move, ‘Alt+R’ rotate, ‘l’ draw line.



PART 3: Bottom-Plate Switch

1) Create a new schematic for the bottom-plate switch. Use the digital gates and transmission gate you have created in the previous parts.



2) Create a symbol for the bottom-plate switch.

Acknowledgement

Thanks to all who contributed to these labs. If you find any errors or have suggestions concerning these labs, please contact Hesham.omran@eng.asu.edu.eg.